BIOINFORMATICS APPLICATIONS NOTE

Vol. 26 no. 8 2010, pages 1125–1126
doi:10.1093/bioinformatics/btq086

Phylogenetics

Advance Access publication March 1, 2010

ScripTree: scripting phylogenetic graphics

François Chevenet[1,2,*], Olivier Croce[3], Maxime Hebrard[2], Richard Christen[3] and Vincent Berry[2]

[1]GEMI, IRD/CNRS UMR 2724, 911 avenue Agropolis, 34394 Montpellier, [2]MAB, LIRMM, CNRS, Université Montpellier, 161, rue Ada, 34392 Montpellier and [3]CNRS UMR 6543 and Université Nice, Centre de Biochimie. Parc Valrose. 06108 Nice, France

Associate Editor: Martin Bishop

ABSTRACT

Summary: There is a large amount of tools for interactive display of phylogenetic trees. However, there is a shortage of tools for the automation of tree rendering. Scripting phylogenetic graphics would enable the saving of graphical analyses involving numerous and complex tree handling operations and would allow the automation of repetitive tasks. ScripTree is a tool intended to fill this gap. It is an interpreter to be used in batch mode. Phylogenetic graphics instructions, related to tree rendering as well as tree annotation, are stored in a text file and processed in a sequential way.

Availability: ScripTree can be used online or downloaded at www.scriptree.org, under the GPL license.

Implementation: ScripTree, written in Tcl/Tk, is a cross-platform application available for Windows and Unix-like systems including OS X. It can be used either as a stand-alone package or included in a bioinformatic pipeline and linked to a HTTP server.

Contact: chevenet@ird.fr

Received on November 17, 2009; revised on February 22, 2010; accepted on February 23, 2010

1 PHYLOGENETIC GRAPHICS

Phylogenetic graphics deals with basic operations on trees (e.g. rooting) and tree rendering processes (e.g. annotation) in the context of large trees and/or collection of trees. Phylogenetic graphics resorts to dynamic information visualization techniques such as ‘focus + context’ magnifying features. Tree annotation consists in the highlighting (coloring, posting of text or symbols) of subtrees or leaf labels according to additional information (e.g. taxonomy, geography, gene function, etc.) related to the entities under study (molecular sequences, species, etc.). Tools like TreeJuxtaposer (Munzner *et al*., 2003), TreeDyn (Chevenet *et al*., 2006), Dendroscope (Huson *et al*., 2007) are examples of tree editors with phylogenetic graphic capabilities. A new challenge in the field is the automation of a graphical analysis encoded as a sequence of operations that precisely describes the way to display and tag trees with additional information. These operations are stored in a script, usable on the same or different datasets. Moreover, scripting is a flexible approach allowing computations to be run either as a local stand-alone process or incorporated within a pipeline and potentially accessible through a web interface. Currently, there is a need for such automation of phylogenetic graphics as we see more and more web sites that provide access to bioinformatic analyses displaying trees, e.g. PhylomeDB (Huerta-Cepas *et al*., 2007), phylogeny.fr (Dereeper *et al*., 2008), PhyloExplorer (Ranwez *et al*., 2009). Yet, few existing tools have scripting capabilities—ATV/Archeopteryx (Zmasek *et al*., 2001), TreeGraph (Muller *et al*., 2004), Ape (Paradis *et al*., 2004), TreeDyn (Chevenet *et al*., 2006), Dendroscope (Huson *et al*., 2007) or ETE (Huerta-Cepas *et al*., 2010). Archeopteryx displays single trees in interactive manipulations. Hence, this is not the tool of choice for automatically rendering tree collections with complex annotations. ETE offers elaborated features for analysis of trees as well as automation and visualization of trees. It is a powerful programmable toolkit, but requires object oriented Python programming skills to annotate trees in an automated way. ScripTree has tree *annotation* features more elaborated than the TreeDyn ones. ScripTree is a higher level interpreter, including numerous and specific annotation commands. It is dedicated to automation and, hence, does not contain a graphical user interface.

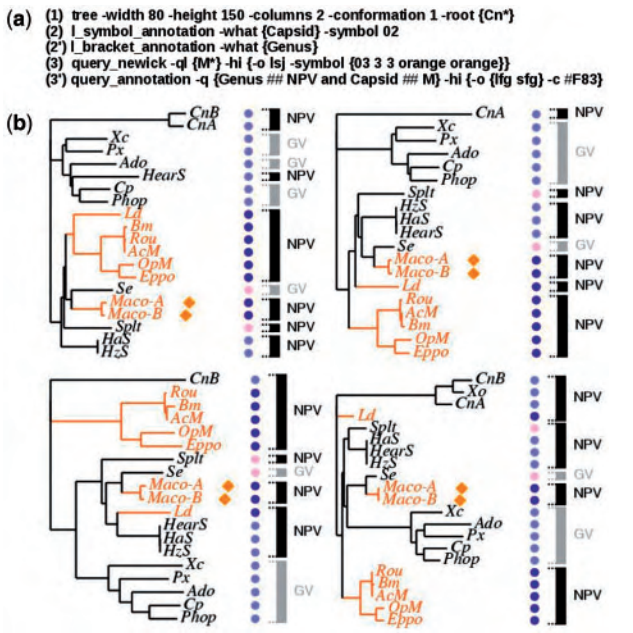
2 SCRIPTREE INPUT/OUTPUT

The basic ScripTree command line is `scriptree -tree file.nwk [options]`. The `-tree` argument refers to a file containing one or more newick strings encoding trees. Without any other specification, ScripTree uses default settings for rendering these trees. Different output file formats are available: PostScript, SVG, PNG and TGF. The SVG format can be displayed by web browsers and edited with drawing programs such as InkScape. The TGF format is useful for an interactive post-processing using the TreeDyn editor. A first optional flag is `-script file.txt` indicating the file containing rendering and annotation commands. The specificity of ScripTree is to take into account additional information into the tree rendering process, indicated by a second optional flag, `-annotation file.txt`. This information is given in a tabular CSV format: annotation variables as columns, and rows related to leaves or internal nodes of the trees. Separating annotations from the tree encoding allows the latter to be compatible with the newick format outputted by common phylogenetic inference programs and the former to be reused on other tree collections.

3 SCRIPTREE COMMANDS

ScripTree commands are divided into three families: edition, projection and identification. In the following, we present examples of commands belonging to these families (Fig. 1a) and apply them

[*]To whom correspondence should be addressed.



```
(a) (1)  tree -width 80 -height 150 -columns 2 -conformation 1 -root {Cn*}
    (2)  l_symbol_annotation -what {Capsid} -symbol 02
    (2') l_bracket_annotation -what {Genus}
    (3)  query_newick -ql {M*} -hi {-o lsj -symbol {03 3 3 orange orange}}
    (3') query_annotation -q {Genus ## NPV and Capsid ## M} -hi {-o {lfg sfg} -c #F83}
```

Fig. 1. Sample of a script (**a**) and image generated by its interpretation by ScripTree (**b**) for four gene trees of 19 virus species (Simon *et al*., 2005). ScripTree's input is (i) a tree file containing four newick strings; (ii) a script file with the commands; (iii) an annotation file with two variables: `Genus` and `Capsid`. The `Genus` variable stores taxonomic information related to two genera, *Nucleopolyhedrovirus* (`NPV`) and *Granulovirus* (`GV`). The `Capsid` variable codes for single (`S`) or multiple (`M`) virion nucleocapsids.

to a collection of four newick strings. Assembling these commands yields Figure 1b.

The *edition* family acts on trees as a whole, specifying: (i) global tree rendering such tree size, leaf label font, organization of a tree collection into rows and columns; (ii) tree manipulations such as branch swapping and rerooting. For instance, command (1) in Figure 1a sets the size of the trees to 80 × 150 pixels, organizes them as a two-by-two matrix, displays them with a rectangular shape accounting for branch lengths (`-conformation 1`) and finally roots the trees at the smallest subtree containing all leaf labels that begin with `Cn` (`CnA` and/or `CnB` depending on the trees).

The *projection* command family allows posting information onto trees. It is organized according to three criteria concerning annotation: (i) its kind (e.g. text, symbol, bracket, arc), (ii) its object (e.g. leaf, edge, subtree) and, (iii) its source, i.e. either the newick string itself (e.g. branch lengths) or the annotation file. Commands (2) and (2′) in Figure 1a put two columns of annotations next to each tree: `l_symbol_annotation` inserts colored circles (`-symbol 02`) in regards of taxa depending on their value for the `Capsid` variable (`-what` flag). The `l_bracket_annotation` command puts brackets (i.e. colored bars) in regards of specific subtrees. Each bracket corresponds to a maximal subtree whose leaves all share a same value for a given variable. Here, the bracket column is related to the `Genus` variable which contains two modalities (`GV` and `NPV`). Several variables can be listed as arguments in a single `-what` flag.

The *identification* commands which enable to highlight only parts of the trees, identified by a query, operate either on the newick string itself (pattern matching allowed) or on the information contained in an annotation file. Each query returns a list of matching labels to which one or several highlighting operations are then applied. Identification commands comprise two parts: selection (`-ql` or `-q` flag) and highlighting (`-hi` flag). For instance, the `query_newick` command (3) in Figure 1a first selects the `Maco-A` and `Maco-B` leaf labels (Fig. 1b), i.e. those matching the `M*` pattern. Then, the `lsj` highlighting operation (`-o` flag), a shortcut for `leaf_symbol_juxtaposition`, posts a symbol in regards of these leaf labels. These symbols can be of different shapes (here `03` for diamonds), size (here 3 × 3 pixels) and color (`orange` for border and fill). Lastly, command (3′) of Figure 1a illustrates how a query command based on annotations selects labels having values `NPV` for the `Genus` variable and `M` for the `Capsid` variable (`-q` flag). Then, the command highlights (`-hi` flag) tree parts by several operations (`-o` flag) switching the leaf label foreground color (`lfg`) and that of corresponding subtrees (`sfg`) to the orange color, here encoded by its hexadecimal value (`-c #F83`).

4 CONCLUSION

ScripTree is a tool for scripting phylogenetic graphics. It allows the management of multiple trees and usual kinds of annotations. It can be used either as a stand-alone package or included in a pipeline and linked to a HTTP server. ScripTree is under continuous development, and suggestions of new functionnalities are welcome.

ACKNOWLEDGEMENTS

We thank V. Guignon and A. Dereeper for comments.

Funding: IRD-SPIRALES 2007; ANR PhylAriane project ANR-08-EMER-011-01; ANR-Biodiversité Aquaparadox project.

Conflict of Interest: none declared.

REFERENCES

Chevenet,F. *et al*. (2006) TreeDyn: towards dynamic graphics and annotations for analyses of trees. *BMC Bioinformatics*, **7**, 439.

Dereeper,A. *et al*. (2008) Phylogeny.fr: robust phylogenetic analysis for the non-specialist. *Nucleic Acids Res.*, **36**, 465–469.

Huerta-Cepas,J. *et al*. (2007) PhylomeDB: a database for genome-wide collections of gene phylogenies. *Nucleic Acids Res.* **36**, 491–496.

Huerta-Cepas,J. *et al*. (2010) ETE: a python environment for tree exploration. *BMC Bioinformatics*, **11**, 24.

Huson,D. *et al*. (2007) Dendroscope: an interactive viewer for large phylogenetic trees. *BMC Bioinformatics*, **8**, 460.

Muller,J. and Muller,K. (2004) TreeGraph: automated drawing of complex tree figures using an extensible tree description format. *Mol. Ecol. Notes*, **4**, 786–788.

Munzner,T. *et al*. (2003) TreeJuxtaposer: scalable tree comparison using focus+context with guaranteed visibility. *ACM Trans. Graphics*, **22**, 453–462.

Paradis,E. *et al*. (2004) APE: analyses of phylogenetics and evolution in R language. *Bioinformatics*, **20**, 289–290.

Ranwez,V. *et al*. (2009) PhyloExplorer: a web server to validate, explore and query phylogenetic trees. *BMC Evol. Biol.*, **9**, 108.

Simon,O. *et al*. (2005) Physical and partial genetic map of Spodoptera frugiperda nucleopolyhedrovirus (SfMNPV) genome. *Virus Genes*, **30**, 403–417.

Zmasek,C.M. *et al*. (2001) ATV: display and manipulation of annotated phylogenetic trees. *Bioinformatics*, **17**, 383–384.